

A Mixtures-of-Experts Framework for Multi-Label Classification

Charmgil Hong
Computer Science Department
University of Pittsburgh
Pittsburgh, PA 15213

Iyad Batal
GE Global Research
San Ramon, CA 94583

Milos Hauskrecht
Computer Science Department
University of Pittsburgh
Pittsburgh, PA 15213

Abstract

We develop a novel probabilistic approach for multi-label classification that is based on the *mixtures-of-experts* architecture combined with recently introduced *conditional tree-structured Bayesian networks*. Our approach captures different input-output relations from multi-label data using the efficient tree-structured classifiers, while the mixtures-of-experts architecture aims to compensate for the tree-structured restrictions and build a more accurate model. We develop and present algorithms for learning the model from data and for performing multi-label predictions on future data instances. Experiments on multiple benchmark datasets demonstrate that our approach achieves highly competitive results and outperforms the existing state-of-the-art multi-label classification methods.

1 INTRODUCTION

Multi-Label Classification (MLC) refers to a classification problem in which data instances are associated with multiple class variables that may reflect different views, functions or components describing the data. MLC naturally arises in many real world problems, such as text categorization (Kazawa et al., 2005; Zhang and Zhou, 2006) where a document may be associated with different topics reflecting its content; semantic scene and video classification (Boutell et al., 2004; Qi et al., 2007a) where different images or videos are assigned to different categories or tagged based on their content; or in genomics where individual genes may be associated with multiple functions (Clare and King, 2001; Zhang and Zhou, 2006).

Formally speaking, the MLC problem is specified by learning a function $h : \mathbb{R}^m \rightarrow \mathbf{Y}$ that maps each data instance, represented by a feature vector $\mathbf{x} = (x_1, \dots, x_m)$, to class assignments, represented by a vector of d binary values $\mathbf{y} = (y_1, \dots, y_d)$, such that

$y_i = \{0, 1\}$ indicates the absence or presence of the i -th class. However, the application of the model in practice raises two important questions: how to define/represent such a function for high-dimensional feature and label spaces; and how to learn this function from data.

The problem of learning MLC classifiers from data has been studied extensively by the machine learning community in recent years. One of the key challenges in solving the problem is how to efficiently model and learn dependences among class variables given the fact that the number of possible class assignments is exponential in d . A simple solution to this is to assume that all class variables Y_i for $i = 1, \dots, d$ are conditionally independent of each other and, hence, learn d functions to predict each class separately (Clare and King, 2001; Boutell et al., 2004). However, this may not suffice when dependences among labels exist. To overcome this limitation, more advanced machine learning methods that model class relations have been proposed. These include two-layer classification models (Godbole and Sarawagi, 2004; Cheng and Hüllermeier, 2009), classifier chains (Read et al., 2009; Zhang and Zhang, 2010; Dembczynski et al., 2010), multi-dimensional Bayesian network classifiers (van der Gaag and de Waal, 2006; Bielza et al., 2011; Antonucci et al., 2013) and output compression and coding methods (Hsu et al., 2009; Tai and Lin, 2010; Zhang and Schneider, 2011, 2012).

However, the above mentioned methods are still rather limited especially when the relations among features and labels become more complex. More specifically, if the relations tend to change across the dataset, these methods may fail to respond with correct classification, since they are designed to capture only one dependence structure from data. For example, in automated image tagging, an object can be tagged as $\{cat, pet\}$ or $\{cat, wild\ animal\}$ according to its context; Similarly, in medical informatics, patients who are suffering from the same disease may receive different sets of medications due to their medical history or allergic reactions. To address such issues, ensemble techniques have been recently adopted to MLC settings (Read et al., 2009; Dembczynski et al., 2010; An-

tonucci et al., 2013). However, these approaches are still limited in that they are relying on randomization for obtaining multiple dependence relations and using simple averaging to make ensemble predictions. As a result, the improvement we could obtain from them is not significant or consistent.

In this paper, we propose and study a new probabilistic MLC approach that attempts to remedy the limitations of the existing approaches. Our approach relies on the *mixtures-of-experts* (ME) framework (Jacobs et al., 1991; Yuksel et al., 2012) and *conditional tree-structured Bayesian network* (CTBN) classifiers proposed recently in (Batal et al., 2013). Briefly, CTBN defines a multi-label classifier that is modeling $P(\mathbf{Y}|\mathbf{X})$ where dependences among class variables for different inputs are modeled by a collection of classifiers (for example, logistic regression models) linked together in directed tree structures. The model comes with a number of computational advantages such as efficient learning, and efficient MAP inference that lets us find the best set of class assignments for a given data instance \mathbf{x} .

A limitation of CTBN is that dependences among class variables are restricted to tree structures which may not reflect all existing dependences in the data. Our new framework based on the ME architecture aims to take advantage of the computational benefits of CTBNs and remedy its limitation by learning and combining multiple CTBNs, where each CTBN can cover a different region of the input space and/or can help to model the different dependences among class variables. We develop and present an EM algorithm for learning the new model from data and an algorithm for making the MAP inferences for predicting class assignments for future data instances.

The rest of the paper is organized as follows. Section 2 formally defines the MLC problem. Section 3 gives summary on related MLC research. Section 4 provides the necessary definitions for ME and CTBN, which are needed to understand our model. Section 5 describes our proposed solution by going over its model representation (Section 5.1) and the supporting algorithms for parameter learning (Section 5.2), structure learning (Section 5.3) and prediction (Section 5.4). Section 6 presents the experimental results and evaluation. Section 7 concludes the paper.

2 PROBLEM DEFINITION

Multi-Label Classification (MLC) is a classification problem in which each data instance is associated with a subset of labels from a set of possible labels L . Let $d = |L|$. We can define d binary class variables Y_1, \dots, Y_d , where the value of Y_i in instance \mathbf{x} indicates whether or not the i -th label in L is present in \mathbf{x} . We are given labeled training data $D = \{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^N$, where $\mathbf{x}^{(n)} = (x_1^{(n)}, \dots, x_m^{(n)})$ is the m -dimensional feature vector of the n -th instance (the input) and

$\mathbf{y}^{(n)} = (y_1^{(n)}, \dots, y_d^{(n)})$ is its d -dimensional class vector (the output). We want to learn a function h that fits D and assigns to each instance, represented by its feature vector, a class vector:

$$h : \mathbb{R}^m \rightarrow \{0, 1\}^d$$

One way to approach this task is to model and learn the *conditional joint distribution* $P(\mathbf{Y}|\mathbf{X})$, where \mathbf{Y} is a random variable for the class vector and \mathbf{X} is a random variable for the feature vector. Assuming the 0-1 loss function, the optimal classifier h^* assigns to each instance \mathbf{x} the maximum a posteriori (MAP) assignment of class variables:

$$\begin{aligned} h^*(\mathbf{x}) &= \arg \max_{\mathbf{y}} P(\mathbf{Y}=\mathbf{y}|\mathbf{X}=\mathbf{x}) \\ &= \arg \max_{y_1, \dots, y_d} P(Y_1=y_1, \dots, Y_d=y_d|\mathbf{X}=\mathbf{x}) \end{aligned} \quad (1)$$

A key challenge for modeling and learning $P(\mathbf{Y}|\mathbf{X})$ from data, as well as for defining the corresponding MAP classifier, is that the number of all possible class assignments for a given \mathbf{x} is exponential in d (there are 2^d different assignments). Our goal is to develop a parametric model that allows us to efficiently model and learn $P(\mathbf{Y}|\mathbf{X})$ from data.

Notation: For notational convenience, we will omit the index superscript $^{(n)}$ when it is not necessary. We may also abbreviate the expressions by omitting variable names; e.g., $P(Y_1 = y_1, \dots, Y_d = y_d|\mathbf{X}=\mathbf{x}) = P(y_1, \dots, y_d|\mathbf{x})$.

3 RELATED RESEARCH

In this section, we review the related research in MLC and outline the main differences from our approach.

Earlier MLC methods ignore the relations between classes and learn to predict each class separately (Clare and King, 2001; Boutell et al., 2004). Zhang and Zhou (2007) presented the multi-label k-nearest neighbor method, which predicts each class label by combining KNN with Bayesian inference. An approach that enriches the feature space by incorporating an intermediate layer of classifiers was proposed by (Godbole and Sarawagi, 2004) and later on by (Cheng and Hüllermeier, 2009). The main drawback of these methods is that class dependences are either not represented at all, or represented indirectly in a limited way.

Several methods have been proposed to probabilistically model MLC. The classifier chains (CC) method (Read et al., 2009) decomposes the relations among the class variables using the chain rule of probability:

$$P(Y_1, \dots, Y_d|\mathbf{X}) = \prod_{i=1}^d P(Y_i|\mathbf{X}, Y_1, \dots, Y_{i-1})$$

Each component in the chain is a classifier that is learned separately by incorporating the (0/1) predictions of preceding classifiers as additional features.

Zhang and Zhang (2010) further studied the influence of the chain order and presented a method to learn an effective ordering from data. The main disadvantage of CC, however, is that they do not perform proper probabilistic inference for classification. Instead, they simply propagate the predictions according to the class order defined by the chain. To overcome this shortcoming, Dembczynski et al. (2010) presented the probabilistic classifier chains method by extending CC to estimate the entire class posterior distribution. However, this method needs to evaluate all possible 2^d label configurations, which greatly limits its applicability.

Other probabilistic MLC methods are based on multi-dimensional Bayesian networks (van der Gaag and de Waal, 2006; Bielza et al., 2011; Antonucci et al., 2013). These methods attempt to build a generative model of $P(\mathbf{X}, \mathbf{Y})$ using a restricted Bayesian network structure, which assumes all class variables are top nodes and all feature variables are their descendants. The limitation of this approach is that it must learn the structure of the full joint distribution over both features and classes, which can be very complex. Besides, it requires all features to be a priori discretized. In contrast, our approach directly learns the conditional distribution $P(\mathbf{Y}|\mathbf{X})$ and takes advantage of modern discriminative classifiers.

An alternative approach for MLC is based on output coding. The idea is to project the output space \mathbf{Y} into a lower dimensional space \mathbf{Y}' , learn to predict \mathbf{Y}' , and then reconstruct the original output from the noisy predictions. Methods that fall into this category use different dimensionality reduction techniques, such as compressed sensing (Hsu et al., 2009), principal component analysis (Tai and Lin, 2010) and canonical correlation analysis (Zhang and Schneider, 2011). The state-of-the-art in output coding utilizes a maximum margin formulation (Zhang and Schneider, 2012) that promotes both discriminative and predictable codes. The limitation of output coding methods is that they can only predict the single “best” output for a given input, and they cannot compute probabilities for different input-output pairs.

Several researchers proposed using ensemble methods for MLC. In general, the objective was to compensate for the restrictions the base MLC models introduce using a set of models and their combinations. Read et al. (2009) presented a simple method that averages the predictions of multiple randomly ordered CCs trained on random subsets of the data. Antonucci et al. (2013) proposed an ensemble of multi-dimensional Bayesian networks combined via simple averaging. In this work, we develop an approach based on the mixtures-of-experts architecture. The difference from the previous work is that our approach optimizes the structures and parameters of base classifiers in a principled way, and that our approach can help to overcome the restriction of the base MLC classifier by modeling better the relations among inputs and their labels, as well as, mutual relations among different labels.

4 PRELIMINARY

The MLC solution we propose in this work combines multiple base MLC classifiers using the *mixtures-of-experts* (ME) (Jacobs et al., 1991) architecture. The base classifiers we use are based on the *conditional tree-structured Bayesian networks* (CTBN) (Batal et al., 2013). To start with, we briefly review the basics of ME and CTBN.

ME is a mixture model that consists of a set of *experts* that are combined via *gating (or switching)* module to represent the conditional distribution $P(y|\mathbf{x})$. The model is defined by the following decomposition:

$$P(y|\mathbf{x}) = \sum_{k=1}^K P(E_k|\mathbf{x})P(y|\mathbf{x}, E_k), \quad (2)$$

where $P(y|\mathbf{x}, E_k)$ is the distribution of outputs defined by the k -th expert E_k and $P(E_k|\mathbf{x})$ is the context sensitive prior of the k -th expert that is implemented by the gating function $g_k(\mathbf{x})$. In general, depending on the choice of the expert model, ME can be used for either regression or classification (Yuksel et al., 2012).

The ME model defines a soft-partitioning of the input space (via the gating module and its functions), on which the K experts represent different input-output relations. ME is especially useful when individual expert models are good in representing local input-output relations but fail to accurately capture the relations for the complete input space. The ability to switch among the experts in different regions of the input space allows to compensate for the limitation of individual experts and improve the overall model and its accuracy. In general, depending on the choice of the expert model, ME can be used for either regression or classification (Yuksel et al., 2012).

ME has been successfully adopted in a range of applications such as handwriting recognition (Ebrahimpour et al., 2009), text classification (Estabrooks and Japkowicz, 2001), climate prediction (Lu, 2006) and bioinformatics (Qi et al., 2007b; Cao et al., 2010). In addition, ME has been vigorously used in time series analysis, including speech recognition (Mossavat et al., 2010), financial forecasting (Weigend and Shi, 2000) and dynamic control systems (Jacobs and Jordan, 1993; Weigend et al., 1995). Recently, ME has been used in social network analysis, in which different social behavior patterns are modeled through mixtures (Gormley and Murphy, 2011).

In this work, we apply the ME approach in context of MLC. We would like to note that although modeling of the joint conditional probability $P(y_1, \dots, y_d|\mathbf{x})$ has been attempted in context of multiple target regression (Jordan and Xu, 1995; Waterhouse, 1997), its application to MLC is new to the best of our knowledge.

In particular, we combine ME with CTBN to model individual experts. CTBN is a recently proposed prob-

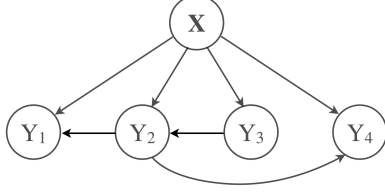


Figure 1: An example of CTBN.

abilistic MLC method that has been shown to be competitive and efficient on a range of domains. CTBN defines $P(\mathbf{Y}|\mathbf{X})$ using a collection of classifiers modeling relations in between features and individual labels that are tied together using a special Bayesian network structure that approximates the dependence relations among the class variables. In modeling of the dependences, it allows each class variable to have at most one other class variable as a parent (without creating a cycle) besides the feature vector \mathbf{X} .

A CTBN T defines the joint distribution of class vector (y_1, \dots, y_d) conditioned on feature vector \mathbf{x} as:

$$P(y_1, \dots, y_d | \mathbf{x}, T) = \prod_{i=1}^d P(y_i | \mathbf{x}, y_{\pi(i, T)}), \quad (3)$$

where $\pi(i, T)$ denotes the parent class of class Y_i in T (by convention, $\pi(i, T) = \{\}$ if Y_i does not have a parent class). For example, the conditional joint distribution of class assignment (y_1, y_2, y_3, y_4) given \mathbf{x} according to the network T in Figure 1 is defined as:

$$\begin{aligned} P(y_1, y_2, y_3, y_4 | \mathbf{x}, T) \\ = P(y_3 | \mathbf{x}) \cdot P(y_2 | \mathbf{x}, y_3) \cdot P(y_1 | \mathbf{x}, y_2) \cdot P(y_4 | \mathbf{x}, y_2) \end{aligned}$$

5 PROPOSED SOLUTION

In this section, we develop a *Multi-Label Mixtures-of-Experts* (ML-ME) model, which uses the ME framework in combination with the CTBN classifiers to improve the classification accuracy of MLC tasks, and develop algorithms for its learning and predictions. Our key motivation is to exploit the divide and conquer principle, which states that a large, complex problem can be decomposed and effectively solved using simpler sub-problems. More specifically, we want to accurately model relations among inputs \mathbf{X} and class variables \mathbf{Y} by learning multiple CTBN models and by improving their predictive ability by combining their outputs. In section 5.1, we describe the mixture defined by the ML-ME model. In section 5.2 through 5.4, we present the learning and prediction algorithms for the ML-ME model.

5.1 REPRESENTATION

By following the definition of ME in Equation (2), ML-ME defines the multivariate posterior distribution of

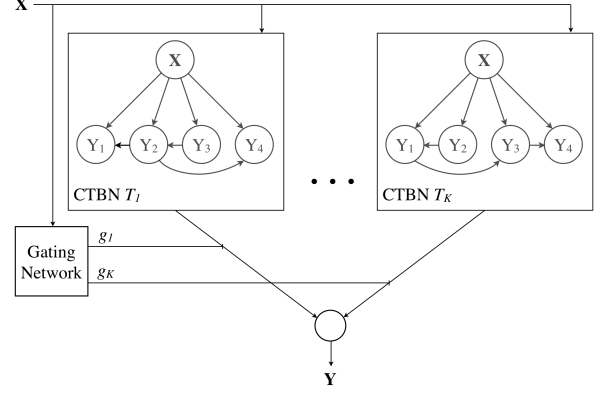


Figure 2: An example of ML-ME.

class vector $\mathbf{y} = (y_1, \dots, y_d)$ as:

$$P(\mathbf{y} | \mathbf{x}) = \sum_{k=1}^K g_k(\mathbf{x}) P(\mathbf{y} | \mathbf{x}, E_k), \quad (4)$$

where $P(\mathbf{y} | \mathbf{x}, E_k)$ is the joint conditional distribution defined by the k -th expert E_k and $g_k(\mathbf{x}) = P(E_k | \mathbf{x})$ is the gating function reflecting how much the k -th expert should contribute to predict classes for input \mathbf{x} . In this work, we model the gating functions for the different experts with the help of the softmax function, which is also known as normalized exponential:

$$g_k(\mathbf{x}) = \frac{\exp(\boldsymbol{\theta}_{G_k} \mathbf{x})}{\sum_{k'=1}^K \exp(\boldsymbol{\theta}_{G_{k'}} \mathbf{x})}, \quad (5)$$

where $\boldsymbol{\theta}_G = \{\boldsymbol{\theta}_{G_k}\}_{k=1}^K$ is a set of the softmax parameters.

To model the different experts and their probabilistic MLC predictions $P(\mathbf{y} | \mathbf{x}, E_k)$ in Equation (4), we rely on the CTBN model introduced in the previous section. Employing multiple CTBN models $T_k : k \in \{1, \dots, K\}$ as experts in the mixture, our ML-ME model defines the joint distribution of class vector \mathbf{y} as:

$$\begin{aligned} P(\mathbf{y} | \mathbf{x}) &= \sum_{k=1}^K g_k(\mathbf{x}) P(\mathbf{y} | \mathbf{x}, T_k) \\ &= \sum_{k=1}^K g_k(\mathbf{x}) \prod_{i=1}^d P(y_i | \mathbf{x}, y_{\pi(i, T_k)}), \end{aligned} \quad (6)$$

Figure 2 depicts an example ML-ME model, which consists of K CTBNs and whose output is probabilistically mixed with the help of the gating network.

Parameters Let $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_G, \boldsymbol{\theta}_T\}$ denote the set of all the parameters of the ML-ME model, where $\boldsymbol{\theta}_G = \{\boldsymbol{\theta}_{G_k}\}_{k=1}^K$ are parameters of the gating model, and $\boldsymbol{\theta}_T = \{\boldsymbol{\theta}_{T_k}\}_{k=1}^K$ are parameters of CTBNs defining individual experts. Since the gating function for each expert is defined by a linear combination of inputs, there are $|\boldsymbol{\theta}_{G_k}| = (m + 1)$ parameters

Table 1: Notations

NOTATION	DESCRIPTION
m	Input (feature) dimensionality
d	Output (class) dimensionality
N	Number of data instances
K	Number of experts in a mixture
T_k	CTBN with index k
$\Theta_T = \{\theta_{T_1}, \dots, \theta_{T_K}\}$	The parameters for CTBNs
$\Theta_G = \{\theta_{G_1}, \dots, \theta_{G_K}\}$	The parameters for a gate

per expert. Consequently, the total number of parameters needed to represent the full gating network is: $|\Theta_G| = K(m+1)$.

The parameterization of a CTBN expert is done by modeling the conditional probability distributions (CPDs) of class variable Y_i conditioned on its parents. For each Y_i , we learn two logistic regression classifier functions according to the parent class: $P(y_i|\mathbf{x}, y_{\pi(i, T_k)} = 0)$ and $P(y_i|\mathbf{x}, y_{\pi(i, T_k)} = 1)$. Let θ_{T_k} denote the parameters for the k -th CTBN expert. Then, for $|\theta_{T_k}| = 2d(m+1)$, since two different classifiers are defined on each class variable. Hence, $|\Theta_T| = 2dK(m+1)$.

Table 1 summarizes the notation and parameterization of our ML-ME model. In summary, the total number of parameters for our model is $K(1+2d)(m+1)$.

5.2 PARAMETER LEARNING FOR FIXED STRUCTURES

In this section, we describe how to learn the parameters of ML-ME, when the structures of individual CTBN experts are known and fixed. We return to the structure learning problem in Section 5.3.

$\Theta = \{\Theta_G, \Theta_T\}$ denotes the set of all the parameters of the ML-ME model. Our objective is to find the parameters that optimize the log-likelihood of the training data:

$$l(D; \Theta) = \sum_{n=1}^N \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)})$$

By substituting the joint probability with the definition of ML-ME (Equation (6)), we obtain:

$$l(D; \Theta) = \sum_{n=1}^N \log \sum_{k=1}^K g_k(\mathbf{x}^{(n)}) P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k) \quad (7)$$

We refer to Equation (7) as the *observed log-likelihood*. However, optimizing this function is very difficult because there is a summation inside the log, which results in a non-convex function. To overcome this difficulty, we instead optimize the *complete log-likelihood* in the expectation-maximization (EM) framework.

The complete log-likelihood is defined by associating each instance $(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})$ with a hidden variable $z^{(n)} \in \{1, \dots, K\}$ indicating to which expert it belongs to:

$$\begin{aligned} l_c(D; \Theta) &= \sum_{n=1}^N \log P(\mathbf{y}^{(n)}, z^{(n)}|\mathbf{x}^{(n)}) \\ &= \sum_{n=1}^N \log \prod_{k=1}^K P(\mathbf{y}^{(n)}, T_k|\mathbf{x}^{(n)}) \mathbb{1}[z^{(n)}=k] \\ &= \sum_{n=1}^N \log \prod_{k=1}^K \left[g_k(\mathbf{x}^{(n)}) P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k) \right] \mathbb{1}[z^{(n)}=k] \\ &= \sum_{n=1}^N \sum_{k=1}^K \mathbb{1}[z^{(n)}=k] \left[\log g_k(\mathbf{x}^{(n)}) P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k) \right], \end{aligned} \quad (8)$$

where $\mathbb{1}[z^{(n)}=k]$ is the indicator function that evaluates to one if the n -th instance belongs to the k -th expert and to zero otherwise.

The EM framework iteratively optimizes the *expected complete log-likelihood* ($E[l_c(D; \Theta)]$), which is always a lower bound of the observed log-likelihood (Dempster et al., 1977). In the *E-step*, the expectation is computed using the current set of parameters; in the *M-step*, the parameters of the model are relearned by maximizing the expected complete log-likelihood. In the following, we explain our parameter learning algorithm by deriving the E-step and M-step for ML-ME.

E-step In the E-step, we compute the expectation of the complete log-likelihood, which reduces to computing the expectation of the hidden variable.

$$\begin{aligned} E[\mathbb{1}[z^{(n)}=k]] &= P(z^{(n)}=k|\mathbf{y}^{(n)}, \mathbf{x}^{(n)}) \\ &= P(T_k|\mathbf{y}^{(n)}, \mathbf{x}^{(n)}) \end{aligned}$$

Notice that this becomes the posterior of the k -th expert given the observation and the current set of parameters. Let $h_k^{(n)}$ denote $P(T_k|\mathbf{y}^{(n)}, \mathbf{x}^{(n)})$. We can write $h_k^{(n)}$ using Bayes rule as:

$$h_k^{(n)} = \frac{g_k(\mathbf{x}^{(n)}) P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k)}{\sum_{k'=1}^K g_{k'}(\mathbf{x}^{(n)}) P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_{k'})} \quad (9)$$

M-step In the M-step, we learn the model parameters $\{\Theta_G, \Theta_T\}$ that maximize the expected complete log-likelihood. Let us first rewrite Equation (8) using $h_k^{(n)}$ and by switching the order of summations:

$$\sum_{k=1}^K \sum_{n=1}^N h_k^{(n)} \log g_k(\mathbf{x}^{(n)}) + h_k^{(n)} \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}, T_k) \quad (10)$$

For the fixed $h_k^{(n)}$ we can decompose Equation (10) into two parts, each of which respectively involves the

gating model parameters Θ_G and the CTBN parameters Θ_T :

$$f_1(D; \Theta_G) = \sum_{k=1}^K \sum_{n=1}^N h_k^{(n)} \log g_k(\mathbf{x}^{(n)})$$

$$f_2(D; \Theta_T) = \sum_{k=1}^K \sum_{n=1}^N h_k^{(n)} \log P(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}, T_k)$$

Notice that, in order to optimize the gate parameters Θ_G , we only need to optimize $f_1(D; \Theta_G)$; whereas to optimize the CTBN parameters Θ_T , we only need to optimize $f_2(D; \Theta_T)$.

We first show how to learn the parameters $\Theta_G = \{\theta_{G_1}, \dots, \theta_{G_K}\}$ of the gating model. We can rewrite $f_1(D; \Theta_G)$ using Equation (5) as:

$$f_1(D; \Theta_G) = \sum_{k=1}^K \sum_{n=1}^N h_k^{(n)} \theta_{G_k} \mathbf{x}^{(n)} - h_k^{(n)} \log \sum_{k'=1}^K \exp(\theta_{G_{k'}} \mathbf{x}^{(n)})$$

Since $f_1(D; \Theta_G)$ is concave in Θ_G , gradient methods are guaranteed to find its optimal solution. The derivative of the log-likelihood with respect to θ_{G_j} is calculated as:

$$\nabla_{\theta_j} f_1(D; \Theta_G) = \sum_{n=1}^N \left\{ h_j^{(n)} - g_j(\mathbf{x}^{(n)}) \right\} \mathbf{x}^{(n)} \quad (11)$$

Note that the derivative becomes zero when $g_j(\mathbf{x}^{(n)}) = P(T_k | \mathbf{x}^{(n)})$ and $h_j^{(n)} = P(T_k | \mathbf{y}^{(n)}, \mathbf{x}^{(n)})$ are equal.

We can solve the optimization of $f_1(D; \Theta_G)$ using any gradient method. However, in practice the dimensionality of the input space can be high which may result in model overfitting. To prevent the overfitting problem, we add L_2 -regularization penalty $R(\Theta_G) = \frac{\lambda}{2} \sum_{k=1}^K \|\theta_{G_k}\|_2^2$ to the optimization function to penalize model complexity. In our experiments, we use the L-BFGS algorithm (Liu and Nocedal, 1989) to optimize this regularized objective function. L-BFGS is a quasi-Newton optimization method that uses a sparse approximation to the inverse Hessian matrix to steer its search through parameter space. The algorithm is known to provide faster convergence rate and be well suited for optimization problems with a large number of variables.

To optimize the parameters of the CTBN experts $\Theta_T = \{\theta_{T_1}, \dots, \theta_{T_K}\}$, we optimize $f_2(D; \Theta_T)$. This optimization decomposes into optimization of parameters of individual CTBN T_k . Note that f_2 is the weighted log-likelihood where $h_k^{(n)}$ serves as the instance weight. Following (Batal et al., 2013), we use the logistic regression model to represent $P(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}, T_k)$ in all our experiments. To prevent overfitting, we apply L_2 -regularized instance-weighted logistic regression to optimize f_2 . Algorithm 1 summarizes our parameter learning algorithm.

Algorithm 1 learn-mixture-parameters

Input: Training data D ; base CTBN experts T_1, \dots, T_K

Output: Model parameters $\{\Theta_G, \Theta_T\}$

```

1: repeat
2:   E-step:
3:   for  $k = 1$  to  $K$ ,  $n = 1$  to  $N$  do
4:     Compute  $h_k^{(n)}$  using Equation (9)
5:   end for
6:   M-step:
7:    $\Theta_G = \arg \max_{\Theta_G} l_1(D; \Theta_G) - R(\Theta_G)$ 
8:   for  $k = 1$  to  $K$  do
9:      $\theta_{T_k} = \arg \max \sum_{n=1}^N h_k^{(n)} \log P(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}, T_k)$ 
10:  end for
11: until convergence

```

5.2.1 Complexity

E-step We compute $h_k^{(n)}$ for each instance on every CTBN expert. This requires $O(md)$ multiplications. Hence, the complexity of a single E-step is $O(KNmd)$.

M-step Computing the derivative in Equation (11) requires $O(mN)$ multiplications, therefore optimizing Θ_G requires $O(mNl)$ operations, where l is the number of L-BFGS steps. To learn Θ_T , we optimize an instance-weighted logistic regression for every node of each of the K experts, which requires learning of $O(Kd)$ logistic regression models.

5.3 STRUCTURE LEARNING

In the previous section, we described the parameter learning of ML-ME by assuming we have fixed the individual CTBN structures. In this section, we present how to automatically learn CTBN structures from data. In a nutshell, we apply a sequential boosting-like heuristic; i.e., on each iteration, we learn a structure that focuses on “hard instances” that previous CTBNs tend to misclassify. In the following, we first describe how to learn a single CTBN structure from instance-weighted data. After that, we describe how to re-weight the instances and incrementally add new structures to the ML-ME model.

5.3.1 Learning a Single CTBN Structure on Weighted Data

To learn the CTBN structure that best approximates weighted data, we find the structure that maximizes the weighted conditional log-likelihood (WCLL) on $\{D, \Omega\}$, where $D = \{\mathbf{x}^{(n)}, \mathbf{y}^{(n)}\}_{n=1}^N$ is the data and $\Omega = \{\omega^{(n)}\}_{n=1}^N$ is the instance weight. Note that we further split D into training data D_{tr} and hold-out data D_h .

Given a CTBN structure T , we train its parameters using D_{tr} , which corresponds to learning instance-weighted logistic regression using D_{tr} and the corresponding instance weights. On the other hand, we use WCLL of D_h to define the score that measures the

quality of T .

$$\begin{aligned} \text{Score}(T) &= \sum_{n \in D_h} \omega^{(n)} \log P(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}, T) \\ &= \sum_{n \in D_h} \sum_{i=1}^d \omega^{(n)} \log P(y_i^{(n)} | \mathbf{x}^{(n)}, y_{\pi(i,T)}^{(n)}) \end{aligned} \quad (12)$$

Below we describe our algorithm for obtaining the CTBN structure that optimizes Equation (12) without having to evaluate all of the exponentially many possible tree structures.

Let us first define a weighted directed graph $G = (V, E)$ as follows:

- There is one vertex V_i for each class variable $Y_i : i \in \{1, \dots, d\}$.
- There is a directed edge $E_{j \rightarrow i}$ from each vertex V_j to each vertex V_i (i.e., G is complete). In addition, each vertex V_i has a self-loop $E_{i \rightarrow i}$.
- The weight of edge $E_{j \rightarrow i}$, denoted as $W_{j \rightarrow i}$, is the WCLL of class Y_i conditioned on \mathbf{X} and Y_j :

$$W_{j \rightarrow i} = \sum_{n \in D_h} \omega^{(n)} \log P(y_i^{(n)} | \mathbf{x}^{(n)}, y_j^{(n)}) \quad (13)$$

- The weight of self-loop $E_{i \rightarrow i}$, denoted as $W_{\phi \rightarrow i}$, is the WCLL of class Y_i conditioned only on \mathbf{X} .

Using the definition of edge weights (Equation (13)) and by switching the order of the summations in Equation (12), we can rewrite the score of T simply as the sum of its edge weights:

$$\text{Score}(T) = \sum_{n=1}^d W_{\pi(i,T) \rightarrow i}$$

Now we have transformed the problem of finding the optimal tree structure into the problem of finding the tree in G that has the maximum sum of edge weights. The solution can be obtained by solving the maximum branching (arborescence) problem (Edmonds, 1967), which finds the maximum weight tree in a weighted directed graph.

5.3.2 Learning Multiple CTBN Structures

In order to obtain multiple, effective CTBN structures for the ML-ME model, we apply the above described algorithm multiple times with different sets of instance weights. We assign the weights such that we give poorly predicted instances higher weights; and give well-predicted instances lower weights.

We start with assigning all instances uniform weights (i.e., all instances are equally important a priori).

$$\omega^{(n)} = 1/N : n = 1, \dots, N$$

Using this initial set of weights, we find the initial CTBN structure T_1 (and its parameters θ_{T_1}) and set

the current model M to be T_1 . We then estimate the prediction error margin $\omega^{(n)} = 1 - P(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}, M)$ for each instance and renormalize such that $\sum_{n=1}^N \omega^{(n)} = 1$. We repeat our structure learning algorithm with $\{\omega^{(n)}\}$ and find another CTBN structure T_2 . After that, we set the current model to be the mixture of T_1 and T_2 and learn the ML-ME parameters $\{\Theta_G, \Theta_T\}$ according to Algorithm 1.

We incrementally add trees to the mixture by repeating this process. To stop the process, we use internal validation approach. Specifically, the data used for learning are split to internal train and test sets. The structure of the trees and parameters are always learned on the internal train set. The quality of the current mixture is evaluated on the internal test set. The mixture growth stops when the log-likelihood on the internal test set for the new mixture is worse than for the previous mixture. The trees included in the previous mixture are then fixed, and the parameters of the mixture are relearned on the full training data.

5.3.3 Complexity

To learn a single CTBN structure, we need to compute edge weights for the complete graph G , which requires estimating $P(Y_i | \mathbf{X}, Y_j)$ for all d^2 pairs of classes. Finding the maximum branching in G can be obtained in $O(d^2)$ using (Tarjan, 1977). To learn K CTBN structures for the mixture, we repeat these steps K times. Therefore, the overall complexity is $O(Kd^2)$ times the complexity of learning logistic regression.

5.4 PREDICTION

In order to make a prediction for a new instance \mathbf{x} , we want to find the MAP assignment of the class variables (see Equation (1)). In general, this requires to evaluate all possible assignments of values to d class variables, which is exponential in d .

One important advantage of the CTBN model is that the MAP inference can be done more efficiently by avoiding blind enumeration of all possible assignments. More specifically, the MAP inference on a CTBN is linear in the number of classes ($O(d)$) when it is implemented using a variant of the max-sum algorithm on a tree structure (Batal et al., 2013).

However, our ML-ME model consists of multiple CTBNs and the MAP solution may, at the end, require enumeration of exponentially many class assignments. To address this problem, we rely on approximate MAP inference. The two commonly applied MAP approximation approaches in the literature are: convex programming relaxation via dual decomposition (Sontag, 2010), and simulated annealing using a Markov chain (Yuan et al., 2004). In this work, we use the latter approach. Briefly, we search the space of all assignments by defining a Markov chain that is induced by local changes to individual class labels. The annealed version of the exploration procedure (Yuan et al., 2004)

is then used to speed up the search. We initialize our MAP algorithm using the following heuristic: first, we identify the MAP assignments for each CTBN in the mixture individually, and after that, we pick the best assignment from among these candidates. We have found this (efficient) heuristic to work very well and often results in the true MAP assignment.

6 EXPERIMENTS

6.1 DATA

We use ten publicly available MLC datasets obtained from different domains including music recognition, semantic image labeling, biology and text classification. Table 2 summarizes the characteristics of the datasets. It shows the number of instances (N), the number of feature variables (m) and the number of class variables (d). In addition, it shows two statistics: 1) label cardinality (LC), which is the average number of labels per instance and 2) distinct label sets (DLS), which is the number of distinct class configurations that appear in the data. Note that for RCV1 datasets we have used the 10 most common labels.

6.2 METHODS

We compare the performance of our proposed method, which we refer to as ML-ME, with the following MLC methods:

- *Binary Relevance (BR)* (Boutell et al., 2004; Clare and King, 2001)
- *Classification with Heterogeneous Features (CHF)* (Godbole and Sarawagi, 2004)
- *Multi-Label k -Nearest Neighbor (MLKNN)* (Zhang and Zhou, 2007)
- *Instance-Based Learning by Logistic Regression (IBLR)* (Cheng and Hüllermeier, 2009)
- *Classifier Chains (CC)* (Read et al., 2009)
- *Ensemble of Classifier Chains (ECC)* (Read et al., 2009)
- *Probabilistic Classifier Chains (PCC)* (Dembczynski et al., 2010)

Table 2: Datasets characteristics

N : number of instances, m : number of features, d : number of classes, LC: label cardinality, DLS: distinct label set, DM: domain

DATASET	N	m	d	LC	DLS	DM
Emotions	593	72	6	1.87	27	music
Yeast	2,417	103	14	4.24	198	biology
Scene	2,407	294	6	1.07	15	image
Image	2,000	135	5	1.24	20	image
Enron	1,702	1,001	53	3.38	753	text
RCV1.sub1	6,000	8,394	10	1.31	69	text
RCV1.sub2	6,000	8,304	10	1.21	70	text
RCV1.sub3	6,000	8,328	10	1.22	74	text
RCV1.sub4	6,000	8,332	10	1.22	79	text
RCV1.sub5	6,000	8,367	10	1.31	76	text

- *Maximum Margin Output Coding (MMOC)* (Zhang and Schneider, 2012)
- *Single CTBN (CTBN)* (Batal et al., 2013)

For all methods, we use the same parameter settings as suggested in the papers that introduced them: For MLKNN and IBLR, we use Euclidean distance to measure similarity of instances and we set the number of nearest neighbors to 10 (Zhang and Zhou, 2007; Cheng and Hüllermeier, 2009); for CC, we set the order of classes to $Y_1 < Y_2, \dots < Y_d$ (Read et al., 2009); for ECC, we use 10 CCs in the ensemble Read et al. (2009); and for MMOC, we set λ (the decoding parameter) to 1 (Zhang and Schneider, 2012). Also note that all of these methods except MMOC are considered as meta-learners because they can work with several base classifiers. To eliminate additional effects that may bias the results, we use L_2 -penalized logistic regression for all of these methods and choose their regularization parameters by cross validation. Lastly, for our ML-ME model, we set the number of mixture components to 5 and we use 150 iterations of simulated annealing for prediction.

6.3 EVALUATION MEASURES

Evaluating the performance of MLC methods is more challenging than that of traditional single-label classification methods. The most appropriate measure is the *exact match accuracy* (EMA), which computes the percentage of instances whose predicted output vectors are exactly the same as their true class vectors (i.e., all classes are predicted correctly). This measure is proper for MLC because it evaluates the success of the method in finding the mode of $P(\mathbf{X}|\mathbf{Y})$ (see Section 2). However, EMA could be too harsh, especially when the output dimensionality is high. An alternative evaluation measure is the *conditional log-likelihood loss* (CLL-loss), which computes the negative conditional log-likelihood of the test instances:

$$CLL\text{-loss} = \sum_{n=1}^N -\log \left(P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}) \right)$$

CLL-loss evaluates how much probability mass is given to the true label vectors (the higher the probability, the smaller the loss). Note that CLL-loss is only defined for probabilistic methods.

Micro F1 and *macro F1* have been used for evaluating MLC methods (Tsoumakas et al., 2009). Micro F1 aggregates the number of true positives, false positives and false negatives for all classes and then calculates the overall F1 score. On the other hand, macro F1 computes the F1 score for each class separately and then averages these scores. Note that both measures are not quite appropriate for MLC because they do not account for the correlations between classes (Dembczynski et al., 2010). However, we report them in our results as they have been used in other MLC literature.

Table 3: Performance of each method on the benchmark datasets in terms of exact match accuracy. Marker */⊗ indicates whether ML-ME is statistically superior/inferior to the compared method (by paired t-test at 0.05 significance level).

EMA	BR	CHF	MLKNN	IBLR	CC	ECC	PCC	MMOC	CTBN	ML-ME
emotions	0.265 *	0.300 *	0.283 *	0.335	0.268 *	0.288 *	0.317	0.332	0.322	0.353
image	0.280 *	0.360 *	0.346 *	0.387 *	0.426 *	0.413 *	0.449 *	0.448 *	0.414 *	0.479
scene	0.541 *	0.605 *	0.629 *	0.644 *	0.632 *	0.658 *	0.666 *	0.664 *	0.625 *	0.711
yeast	0.151 *	0.163 *	0.179 *	0.204 *	0.193 *	0.204 *	0.230	0.219 *	0.192 *	0.244
enron	0.164 *	0.170 *	0.078 *	0.163 *	0.173 *	0.180 *	-	-	0.167 *	0.196
rcv1.sub1	0.334 *	0.357 *	0.205 *	0.279 *	0.429 *	0.410 *	0.432 *	-	0.441 *	0.470
rcv1.sub2	0.439 *	0.465 *	0.288 *	0.417 *	0.516 *	0.509 *	0.523 *	-	0.531	0.536
rcv1.sub3	0.466 *	0.486 *	0.327 *	0.446 *	0.539 *	0.539 *	0.548 *	-	0.560 *	0.565
rcv1.sub4	0.510 *	0.531 *	0.354 *	0.491 *	0.579 *	0.569 *	0.588	-	0.592	0.592
rcv1.sub5	0.439 *	0.456 *	0.276 *	0.411 *	0.497 *	0.494 *	0.519 *	-	0.539	0.538

Table 4: Performance of each method on the benchmark datasets in terms of conditional log-likelihood loss. Marker */⊗ indicates whether ML-ME is statistically superior/inferior to the compared method (by paired t-test at 0.05 significance level).

CLL-Loss	BR	CHF	MLKNN	IBLR	CC	PCC	CTBN	ML-ME
emotions	153.5 *	147.5 *	151.7 *	143.0 *	169.6 *	134.9	147.4 *	131.3
image	432.5 *	415.9 *	425.3 *	395.6 *	480.3 *	354.7	388.4 *	339.2
scene	344.7 *	318.4 *	310.9 *	283.9 *	395.0 *	258.9 *	306.3 *	234.6
yeast	1500.3 *	1491.7 *	1464.9 *	1434.2 *	2303.8 *	932.1 ⊗	1097.0 *	989.5
enron	1290.0 *	1272.5 *	1301.2 *	1287.4 *	1293.5 *	-	1437.9 *	1213.4
rcv1.sub1	1443.8 *	2144.2 *	1873.7 *	1379.5 *	1701.3 *	1034.3 *	962.7	980.9
rcv1.sub2	1207.4 *	2223.6 *	1687.8 *	1172.6 *	1398.8 *	923.0 *	893.5 *	858.1
rcv1.sub3	1207.4 *	2156.0 *	1674.6 *	1168.2 *	1500.5 *	896.7 *	939.7 *	840.1
rcv1.sub4	1072.9 *	1759.9 *	1532.9 *	1034.8 *	1282.1 *	823.0 *	790.7 *	770.7
rcv1.sub5	1267.0 *	2283.6 *	1795.5 *	1234.7 *	1422.0 *	1009.0 *	924.0 *	898.4

6.4 RESULTS

Tables 3, 4, 5 and 6 show the performance of all methods in terms of EMA, CLL-loss, micro F1 and macro F1, respectively. All results are obtained using *ten-fold cross validation*. To evaluate the statistical significance in the performance measure differences, we use paired t-test at 0.05 significance level. We use markers */⊗ to indicate whether ML-ME is statistically superior/inferior to the compared method.

Note that we report the results of MMOC only on four datasets (emotions, image, scene and yeast) because it did not finish on the rest of the datasets (MMOC did not finish one round of the learning within 24 hours). Also, PCC did not finish on the enron dataset because it has to evaluate all 2^{53} possible class assignments, which is clearly infeasible. Lastly, we do not report CLL-loss for MMOC and ECC because they do not compute the probabilistic score for a giving class assignment.

In terms of EMA (Table 3), ML-ME clearly outperforms the other methods on most datasets. For example, ML-ME is significantly better than BR, CHF, MLKNN, CC and ECC on all ten datasets, significantly better than IBLR on nine datasets, significantly better than PCC on six datasets and significantly better than MMOC on three datasets. In addition, ML-ME shows significant improvement over a single CTBN on six datasets, which demonstrates the ability of the mixture to compensate for the tree structure restric-

tion of a single CTBN.

In terms of CLL-loss (Table 4), ML-ME again shows consistent improvement over the other methods. This is because ML-ME is learned to optimize the log-likelihood of the data and its prediction is based on exact MAP inference. Also note that ML-ME shows a consistent improvement over CTBN because combining multiple CTBNs allows us to account for different relations in the data and, hence, improves the generalization of the model.

Lastly, Table 5 and 6 show that ML-ME is also very competitive in terms of micro and macro F1 scores, although optimizing them was not our main objective.

In summary, the experimental results show that our ME-ML method with CTBN experts is able to outperform or match the existing state-of-the-art methods across a broad range of benchmark MLC datasets. We attribute this performance to the ability of CTBN mixtures to simultaneously compensate for the restricted dependences modeled by an individual CTBN, and for its ability to fit better the different regions of the input space with new expert models.

7 CONCLUSION

In this paper, we have proposed a new probabilistic approach for the multi-label classification problem. Our approach models different input-output relations using

Table 5: Performance of each method on the benchmark datasets in terms of micro F1.

Marker */⊗ indicates whether ML-ME is statistically superior/inferior to the compared method (by paired t-test at 0.05 significance level).

MICRO F1	BR	CHF	MLKNN	IBLR	CC	ECC	PCC	MMOC	CTBN	ML-ME
emotions	0.645 *	0.672	0.656 *	0.692	0.621 *	0.652 *	0.664	0.687	0.678	0.694
image	0.479 *	0.541 *	0.504 *	0.573	0.550 *	0.563 *	0.565 *	0.572 *	0.561 *	0.597
scene	0.696 *	0.722 *	0.736 *	0.758	0.697 *	0.724 *	0.722 *	0.711 *	0.717 *	0.762
yeast	0.635	0.637	0.646	0.661 ⊗	0.628	0.631	0.645	0.651	0.631	0.640
enron	0.551 *	0.569	0.450 *	0.566	0.577	0.583 ⊗	-	-	0.552	0.565
rcv1.sub1	0.503 *	0.516 *	0.257 *	0.459 *	0.511 *	0.525	0.510 *	-	0.512 *	0.533
rcv1.sub2	0.568 *	0.584 *	0.317 *	0.546 *	0.586	0.589	0.588	-	0.591	0.590
rcv1.sub3	0.576 *	0.592 *	0.364 *	0.564 *	0.594	0.610 ⊗	0.594	-	0.596 *	0.602
rcv1.sub4	0.622 *	0.637	0.404 *	0.606 *	0.640	0.646 ⊗	0.644 ⊗	-	0.638	0.634
rcv1.sub5	0.582 *	0.597	0.314 *	0.566 *	0.595	0.603 ⊗	0.600 ⊗	-	0.598	0.595

Table 6: Performance of each method on the benchmark datasets in terms of macro F1.

Marker */⊗ indicates whether ML-ME is statistically superior/inferior to the compared method (by paired t-test at 0.05 significance level).

MACRO F1	BR	CHF	MLKNN	IBLR	CC	ECC	PCC	MMOC	CTBN	ML-ME
emotions	0.632 *	0.667	0.656 *	0.690	0.620 *	0.643 *	0.659 *	0.679	0.670	0.692
image	0.486 *	0.546 *	0.516 *	0.581	0.562 *	0.571 *	0.575 *	0.578 *	0.572 *	0.605
scene	0.703 *	0.730 *	0.743 *	0.765	0.709 *	0.740 *	0.729 *	0.721 *	0.728 *	0.772
yeast	0.457 *	0.461	0.478	0.498 ⊗	0.467	0.477	0.486	0.473	0.467	0.472
enron	0.478	0.479	0.411 *	0.475	0.484	0.482	-	-	0.470	0.483
rcv1.sub1	0.495 *	0.511 *	0.273 *	0.463 *	0.506 *	0.516	0.504 *	-	0.507 *	0.525
rcv1.sub2	0.503 *	0.526 *	0.264 *	0.475 *	0.531	0.539	0.531	-	0.536	0.536
rcv1.sub3	0.513 *	0.536 *	0.278 *	0.497 *	0.547	0.558 ⊗	0.548	-	0.543	0.548
rcv1.sub4	0.499 *	0.519	0.269 *	0.477 *	0.534	0.540 ⊗	0.534 ⊗	-	0.526	0.522
rcv1.sub5	0.500 *	0.526	0.257 *	0.487 *	0.536 ⊗	0.538 ⊗	0.534 ⊗	-	0.536 ⊗	0.528

conditional tree-structured Bayesian networks, while the mixtures-of-experts architecture aims to compensate for the tree-structured restrictions and as a result achieves a more accurate model. We have formulated and developed the algorithms for learning the model from data and for performing multi-label predictions on future data instances. Our experiments on a broad range of datasets showed that our approach outperforms several state-of-the-art methods and produces more reliable probabilistic estimates.

References

- Alessandro Antonucci, Giorgio Corani, Denis Deratani Mauá, and Sandra Gabaglio. An ensemble of bayesian networks for multilabel classification. In *IJCAI*, pages 1220–1225, 2013.
- Iyad Batal, Charmgil Hong, and Milos Hauskrecht. An efficient probabilistic framework for multi-dimensional classification. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM ’13, pages 2417–2422. ACM, 2013.
- C. Bielza, G. Li, and P. Larrañaga. Multi-dimensional classification with bayesian networks. *International Journal of Approximate Reasoning*, 52(6):705 – 727, 2011.
- Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757 – 1771, 2004.
- Kim-Anh L Cao, Emmanuelle Meugnier, and Geoffrey J. McLachlan. Integrative mixture of experts to combine clinical factors and gene markers. *Bioinformatics*, 26(9): 1192–1198, 2010.
- Weiwei Cheng and Eyke Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.
- Amanda Clare and Ross D. King. Knowledge discovery in multi-label phenotype data. In *In: Lecture Notes in Computer Science*, pages 42–53. Springer, 2001.
- Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 279–286. Omnipress, 2010.
- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39: 1–38, 1977.
- Reza Ebrahimpour, Mohammad R. Moradian, Alireza Esmkhani, and Farzad M. Jafarlou. Recognition of persian handwritten digits using characterization loci and mixture of experts. *JDCTA*, 3(3):42–46, 2009.
- Jack Edmonds. Optimum branchings. *Research of the National Bureau of Standards*, 71B:233–240, 1967.
- Andrew Estabrooks and Nathalie Japkowicz. A mixture-of-experts framework for text classification. In *Proceedings of the 2001 Workshop on Computational Natural Language Learning - Volume 7*, ConLL ’01, pages 9:1–9:8, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics.
- Shantanu Godbole and Sunita Sarawagi. Discriminative

- methods for multi-labeled classification. In *PAKDD'04*, pages 22–30, 2004.
- Isobel Claire Gormley and Thomas Brendan Murphy. *Mixture of Experts Modelling with Social Science Applications*, pages 101–121. John Wiley & Sons, Ltd, 2011.
- Daniel Hsu, Sham Kakade, John Langford, and Tong Zhang. Multi-label prediction via compressed sensing. In *NIPS*, pages 772–780, 2009.
- Robert A. Jacobs and Michael I. Jordan. Learning piecewise control strategies in a modular neural network architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(2):337–345, 1993.
- Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Comput.*, 3(1):79–87, March 1991. ISSN 0899-7667.
- Michael I. Jordan and Lei Xu. Convergence results for the em approach to mixtures of experts architectures. *Neural Networks*, 8(9):1409–1431, 1995.
- Hideto Kazawa, Tomonori Izumitani, Hirotohi Taira, and Eisaku Maeda. Maximal margin labeling for multi-topic text categorization. In *Advances in Neural Information Processing Systems 17*, pages 649–656. MIT Press, 2005.
- D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528, December 1989.
- Zhiwu Lu. A regularized minimum cross-entropy algorithm on mixtures of experts for time series prediction and curve detection. *Pattern Recogn. Lett.*, 27(9):947–955, July 2006. ISSN 0167-8655.
- S. I. Mossavat, O. Amft, B. De Vries, Petko Petkov, and W. Bastiaan Kleijn. A bayesian hierarchical mixture of experts approach to estimate speech quality. In *2010 2nd International Workshop on Quality of Multimedia Experience*, volume QoMEX 2010 - Proceedings, pages 200–205, 2010.
- Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, Tao Mei, and Hong-Jiang Zhang. Correlative multi-label video annotation. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 17–26. ACM, 2007a.
- Y. Qi, J. Klein-Seetharaman, and Z. Bar-Joseph. A mixture of feature experts approach for protein-protein interaction prediction. *BMC bioinformatics*, 8(Suppl 10): S6, 2007b. ISSN 1471-2105.
- Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 254–269. Springer-Verlag, 2009.
- David Sontag. *Approximate Inference in Graphical Models using LP Relaxations*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2010.
- Farbound Tai and Hsuan-Tien Lin. Multi-label classification with principle label space transformation. In *Proceedings of the 2nd International Workshop on Multi-Label Learning*, 2010.
- Robert Endre Tarjan. Finding optimum branchings. *Networks*, 7(1):25–35, 1977.
- Grigorios Tsoumakas, Min-Ling Zhang, and Zhi-Hua Zhou. Learning from multi-label data. ECML PKDD Tutorial, 2009.
- Linda C. van der Gaag and Peter R. de Waal. Multi-dimensional bayesian network classifiers. In *Probabilistic Graphical Models*, pages 107–114, 2006.
- Steven Richard Waterhouse. *Classification and Regression Using Mixtures of Experts*. PhD thesis, University of Cambridge, Department of Engineering, 1997.
- A. S. Weigend, M. Mangeas, and A. N. Srivastava. Non-linear gated experts for time series: Discovering regimes and avoiding overfitting. *International Journal of Neural Systems*, 6:373–399, 1995.
- Andreas S. Weigend and Shanming Shi. Predicting daily probability distributions of S&P500 returns. *Journal of Forecasting*, 19(4), July 2000.
- Changhe Yuan, Tsai-Ching Lu, and Marek J. Druzdzel. Annealed map. In *UAI*, pages 628–635. AUAI Press, 2004.
- Seniha Esen Yuksel, Joseph N. Wilson, and Paul D. Gader. Twenty years of mixture of experts. *IEEE Trans. Neural Netw. Learning Syst.*, 23(8):1177–1193, 2012.
- Min-Ling Zhang and Kun Zhang. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 999–1008. ACM, 2010.
- Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 2006.
- Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recogn.*, 40(7):2038–2048, July 2007.
- Yi Zhang and Jeff Schneider. Multi-label output codes using canonical correlation analysis. In *AISTATS 2011*, 2011.
- Yi Zhang and Jeff Schneider. Maximum margin output coding. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 1575–1582. Omnipress, 2012.